process of categorization. Classifiers include connectors, datatypes, interfaces, models, and signals. Classifiers are described using classifier descriptors including connector descriptors, datatype descriptors, interface descriptors, model descriptors, and signal descriptors.

[**0044**] For the purposes of the present invention, the term "command" refers to a component that performs a specific action. In addition a command must describe the input data it expects and the output data it produces.

[**0045**] For the purposes of the present invention, the term "component integration engine" or "CIE" refers an application that is able to assemble software applications directly from the description of the software application. A component integration engine provides a mechanism for using the meta-implementation layer to construct software applications in a consistent and effective manner through simple integration techniques without requiring lower-level programming language implementation. A component integration engine is a software engine that combines software components through metadata.

[**0046**] For the purposes of the present invention, the term "machine readable medium" refers to any medium or media on which a software program or data for a software program may be stored for use by a computer system. Examples of data machine readable media include such as floppy disks, Zip™ disks, CD-ROM, CD-R, CD-RW, DVD, DVD-R, flash memory, hard disks, optical disks, etc. The meta-implementation layer and/or component integration engine or any part of the meta-implementation layer or component integration engine of the present invention may be stored on one or more machine readable media that together effectively act similarly to single machine readable medium. Two or more machine readable media acting similarly to single machine readable medium may be referred to as a "machine readable medium" for the purposes of the present invention.

[**0047**] For the purposes of the present invention, the term "software component" or "component" refers to a binary object or program that performs a specific function and is designed in such a way to easily operate with other components and applications.

[**0048**] For the purposes of the present invention, the term "constraint" refers to a limit or restriction in a metamodel. Several types of constraints exist in metamodels. Occurrence constraints limit the number of times an instance can occur in the model. Occurrences cannot be negative. Value constraints limit the values allowed for an attribute. Access constraints restrict the use of a feature to users who hold the required credentials. Some examples of occurrence constraints are: "the box model can contain a maximum of twelve instances,""the box model must contain at least one can instance,""the car model can hold from zero to seven passenger instances," etc. Some examples of value constraints are: "the 'apple' model 'color' attribute is of datatype string and must be one of: ['Red,''Green,''Yellow'], "the 'test' model 'percentage right' value is of datatype number and must be between 0 and 100,""the 'cat' model 'owner name' attribute value is of datatype string and must be less than 60 characters long," etc. Some examples of access constraints: The "apple" model "color" attribute is restricted to users in the "grocer" group. The "test" model "percentage right" value can only be written to if the user is in the "teacher" group.

[**0049**] For the purposes of the present invention, the term "descriptor" refers to an interface implemented by classes that describe an implementation. The descriptor contains the descriptions of features and functionality allowed and required in an implementation. A descriptor is a specific type of metadata.

[**0050**] For the purposes of the present invention, the term "failure" refers to a software operation being unable to successfully complete, breaking out of the current execution and rising up the stack until the failure can be handled. Failures are broken into several categories: A "functional error" occurs when an operation should succeed, but an unusual or infrequent condition in the system prevented success. A "data error" occurs when improper or missing data is provided to an operation which requires that data in order to succeed. A "data bug" occurs when data is provided that is correct according to the rules for the data but the program did not anticipate and cannot handle that data. A "functional bug" describes an operation that successfully completes even though it completes the incorrect operation. Functional bugs are not always automatically detected during execution. Some functional bugs are detected by post-condition constraints.

[**0051**] For the purposes of the present invention, the term "feature descriptor" refers to a part of a metamodel that describes each feature of the model described by the metamodel. Feature descriptors for operations are referred to as "operation descriptors". Feature descriptors for attributes are referred to as "attribute descriptors". Feature descriptors for operation parameters are referred to as "parameter descriptors". Feature descriptors for constraints are referred to as "constraint descriptors". Feature descriptors for constructors are referred to as "constructor descriptors". Feature descriptors for destructor are referred to as "destructor descriptors". Feature descriptors for failures are referred to as "failure descriptors". Feature descriptors for signals are referred to as "signal descriptors". Each feature descriptor also includes name, description, and display name attributes. The name attribute is used to identify the name of the feature being described. The description attribute is used to describe the usefulness and purpose of the feature. The display name attribute is used to present a human readable name for the feature and may or may not be different from the value assigned to the name attribute.

[**0052**] For the purposes of the present invention, the term "generalization/specialization relationship" refers to a representation of the modeling processes of abstraction and categorization. A model specializes another model of the present invention by extending it. A model generalizes several other models by serving as a base model to those other models. Interfaces also participate in generalization/specialization relationships. Models specialize interfaces by implementing them in a specific way. Interfaces generalize models that share certain characteristics to in order to allow for polymorphic functionality.

[**0053**] For the purposes of the present invention, the term "implementation" refers to a structure that provides a mechanism to execute of all operations (including methods, signals, etc.) described in a descriptor, a mechanism to hold all static data described in a descriptor (including attributes, parameters, etc.), and a mechanism for creating instances of the type described by the descriptor. The instances created